Lossy Compression of Packet Classifiers

Ori Rottenstreich
Princeton University
orir@cs.princeton.edu

J'anos Tapolcai
MTA-BME Future Internet Research Group,
Budapest University of Technology and Economics (BME)
tapolcai@tmit.bme.hu

ABSTRACT

Packet classification is a building block in many network services such as routing, filtering, intrusion detection, accounting, monitoring, load-balancing and policy enforcement. Compression has gained attention recently as a way to deal with the expected increase of classifiers size. Typically, compression schemes try to reduce a classifier size while keeping it semantically-equivalent to its original form. Inspired by the advantages of popular compression schemes (e.g. JPEG and MPEG), we study in this paper the applicability of *lossy compression* to create packet classifiers requiring *less* memory than optimal semantically-equivalent representations. Our objective is to find a limited-size classifier that can correctly classify a high portion of the traffic so that it can be implemented in commodity switches with classification modules of a given size. We develop optimal dynamic programming based algorithms for several versions of the problem and describe how a small amount of traffic that cannot be classified can be easily treated, especially in software-defined networks. We generalize our solutions for a wide range of classifiers with different similarity metrics. We evaluate their performance on real classifiers and traffic traces and show that in some cases we can reduce a classifier size by orders of magnitude while still classifying almost all traffic correctly.

1. INTRODUCTION

Packet classification is a core function behind many network services such as routing, filtering, intrusion detection, accounting, monitoring, load-balancing and policy enforcement. In recent years there has been a rapid growth in the size of classifiers and routing tables resulting in a scalability problem [1, 2]. In commodity switches, classification often relies on TCAMs. TCAMs are used to perform very-high-speed, hardware accelerated table lookups for IP prefixes but are expensive, power hungry, and often of a limited size [3–5].

Measurements show that Internet traffic tends to follow the Zipf distribution [6] and that a large portion of the traffic comes from a small number of flows [7,8]. Accordingly traffic matches the classification rules in a biased distribution such that some of the classifier information is very seldom useful. This observation has motivated adapting cache-based scheme for classifiers [9, 10] similar to those used in CPU caching.

Huffman coding [11] and the Lempel-Ziv-Welch algorithm [12] are well known lossless compression schemes. Lossless compression of packet classifiers has been deeply investigated in the last decades [13–19]. The ORTC algorithm [20] achieves an optimal representation with a minimal number of prefix rules. Ideally lossless compressions can reach the entropy bounds [21], but for a practical memory-efficient classifier in commodity switches we need to go beyond these bounds.

Lossy compression is a methodology for achieving higher compression ratios at the cost of losing some information about the represented object. Lossy representations can be smaller than the lower bounds for a lossless compression. Implementations of lossy compression schemes can be found in popular standards and applications such as JPEG (for images), MPEG (for videos) and MP3 (for audio) [22–24].

In the paper, we study the applicability of *lossy compression for packet classifiers*. We would like to implement within a limited-size module a classifier that is similar under different metrics to the required one while satisfying various constraints. To the best of our knowledge, this is the first time that lossy compression techniques have been suggested in the context of packet classifiers. In the main scheme of our approach a unique action must be returned for all packets that cannot be classified due to the lossy representation of the classifier. For the unclassified packets we can then calculate the classification in an alternative slower module. We answer the question how to optimally determine the content of the fast classifier to maximize the traffic portion it can classify. We also provide a toy scheme that serves as a baseline for better understanding of the main scheme. In this toy scheme false classifications are allowed and can occur for a portion of the traffic. We define optimization problems for finding limited-size encodings for the two schemes. We generalize the problems and present algorithms that optimally solve them.

To illustrate the lossy compression schemes, we consider a classifier with five rules presented in Fig. 1a. An equivalent representation with the minimal number of prefix rules, four in this case, is the ORTC representation. It appears in Fig. 1b and maps the corresponding action to every header with a width of $W = 3$ bits. Assume that the $2^W = 8$ headers appear according to a uniform distribution U and that only $n = 3$ rules are allowed in a limited-size memory.

The first (toy) scheme, called *Approximate Classification* (illustrated in Fig. 1c), uses only three rules to classify correctly 7 of the 8 possible headers. With this encoding, only the header 101, which appears with probability 0.125, is mapped to an incorrect action of 1 instead of to the action 4. We say that this scheme achieves an approximation ratio of $7/8 = 0.875$. In the second (main) scheme, *Cached Classification* (in Fig. 1d), all headers that cannot be classified correctly are indicated by the special action '?'. With the same number of rules (three), this scheme correctly classifies six of the eight headers, obtaining an approximation ratio

978-1-4673-6633-5/15/$31.00

prefix/length	action
100/3	3
101/3	4
01/2	2
00/2	1
11/2	1

(a) LPM Classifier, 5 rules

prefix/length	action
100/3	3
101/3	4
01/2	2
-/0	1

(b) ORTC exact representation [20], 4 rules

prefix/length	action
100/3	3
01/2	2
-/0	1

(c) Approximate Classification, approximation ratio of 0.875 for $n = 3$ rules

prefix/length	action
10/2	'?'
01/2	2
-/0	1

(d) Cached Classification, approximation ratio of 0.75 for $n = 3$ rules

Figure 1: Illustration of the suggested encodings for limited size classifiers with a width of $W = 3$ bits and $2^W = 8$ uniformly-distributed headers. (a) shows an LPM-based classifier with 5 rules and (b) presents its ORTC compressed representation [20] with 4 rules. (c) describes an *Approximate Classification* encoding of the classifier given $n = 3$ rules. All headers besides 101 are classified correctly, yielding an approximation ratio of $7/8 = 0.875$. (d) illustrates the *Cached Classification* encoding given $n = 3$ rules. The headers 100, 101 are mapped to the unique action '?' (unclassified) while all other headers are classified correctly, yielding an approximation ratio of $6/8 = 0.75$.

of $6/8 = 0.75$, and gives a special indication for the two headers (100 and 101) that it does not classify correctly.

The suggested solutions do not directly rely on the TCAM architecture. We consider encodings of classifiers composed of prefix rules which are common especially in the context of longest prefix match (LPM) classifiers where in the case of several matching rules, a priority is given to the most specific one. Indeed, our approach can be useful to deal with limited number of allowed rules in additional architectures that implement LPM other than TCAM, e.g. in the BST (Binary Search Tree) memory of Intel's FlexPipe architecture that supports LPM with up to 64K rules [25].

Paper outline: In Section 2 we explain the terminology of the paper. Next, in Section 3 we define the two schemes of Approximate Classification and Cached Classification and point on their important properties in Section 4. Then in Section 5 and Section 6, respectively, we present optimal algorithms for the two problems. Generalizations of the approach are described in Section 7 for classifiers with numerical classification values and for two-dimensional classifiers. In Section 8 we discuss ways for coping with incorrect classifications and non-classified traffic as well as the applicability of the approach to a wide range of problems. Experimental results that demonstrate the potential of the lossy compression approach are given in Section 9. Related work can be found in Section 10. Proofs of the main results appear in the Appendix.

2. MODEL AND NOTATION

We first formally define the terminology of this paper.

DEFINITION 1. *A* **packet header** $x = (x_1, \cdots, x_W) \in \{0,1\}^W$ *is defined as a W-bit string that serves as an input to the classification process.*

In the main part of the paper we assume a simple case in which the classification is performed on a single field, e.g. the source or the destination IP address. Note that the typical values for W are 32 for IPv4 addresses, and 128 for IPv6. We later discuss a more general case.

DEFINITION 2. *A* **prefix rule**, *denoted by $S \to a$, is defined as a string $S = s_1 \dots s_k \in \{0,1\}^k$ of length $k \leq W$ associated with an action a among a set of actions $\mathcal{A}$. A packet header $x = x_1 \dots x_W$ is said to* **match** *a rule S, if and only if for all $i \in [1,k]$, $s_i = x_i$.*

DEFINITION 3. *A* **classification function** *defines the mapping of each header to an* **action** *$a \in \mathcal{A}$.*

DEFINITION 4. *A* **prefix classifier** $C^\phi = (S^1 \to a^1, ..., S^n \to a^n)$ *is an ordered set of prefix rules. It encodes a classification function ϕ such that for any packet header $x \in \{0,1\}^W$ we have $\phi(x) = a^j$, where $S^j \to a^j$ is the prefix rule with the longest length that matches x. We also refer to a classifier as an encoding.*

To guarantee that the classification function of every encoding is well defined, we assume a default action $a^- \in \mathcal{A}$ to which headers that do not match any of the rules are mapped. We refer to a classifier that implements a function ϕ by C^ϕ.

DEFINITION 5. *We assume during the classification the packets appear according to a* **header distribution** *P, where p_x denotes the probability of a header x.*

A classification function α and a header distribution P are the input of the problems discussed in this paper. The classification function can be described by a corresponding classifier C^α. In practice the exact header distribution might be unknown and instead it is only estimated by traffic sampling. Eventually, this estimation might result in some performance degradation, which is examined in the simulation section.

For a given classification function α we say that a classifier is an exact representation if it implements exactly the same function. A classifier is an exact representation of only one classification function. Note that the same classification function can be represented by several classifiers, possibly with different number of rules. As mentioned, an exact representation with the smallest number of prefix rules can be computed with the ORTC algorithm [20]. For a given α, we denote this minimal number of rules by n_0.

This paper studies a scenario in which the classification module can store fewer rules than the minimal number required for an exact representation. For instance, in the example from Fig. 1, with fewer than four rules we cannot guarantee correct classification for all inputs.

3. OPTIMIZATION PROBLEMS

3.1 Formal Definitions

We first define a metric that estimates the similarity of a classifier to a given classification function.

DEFINITION 6. *Let α be a classification function and P a header distribution. Let ϕ be a second classification function.*

The **approximation ratio** *of a classifier that implements ϕ, denoted by $R_P(\alpha, \phi)$, is the probability of a header drawn according to the header distribution P to be classified to the same action in α and in ϕ. Formally,*

$$R_P(\alpha, \phi) = \sum_{x \in \{0,1\}^W | \alpha(x) = \phi(x)} p_x.$$

In the first optimization problem, we would like to find an encoding with a limited number of rules that achieves a maximal approximation ratio.

PROBLEM 1 (APPROXIMATE CLASSIFICATION). *For a given classification function α, a header distribution P and a given number of prefix rules n, find a classifier C^ϕ with at most n rules that obtains a maximal approximation ratio*

$$\mathcal{G}(n, \alpha, P) = \max_{C^\phi, |C^\phi| \leq n} R_P(\alpha, \phi).$$

Note that in the Approximate Classification problem a header that is not classified correctly can be mapped to an arbitrary action.

We denote by $\mathcal{G}(n, \alpha, P)$ the optimal (maximal) value of the above function. We refer to a legal encoding (with at most n rules) that obtains this approximation ratio as an approximation-optimal encoding. We also define the **error ratio** of a classifier that implements ϕ as $1 - R_P(\alpha, \phi)$.

In typical network applications, incorrect classification can be harmful. It can be more useful to avoid any wrongly classified packets by leaving some packets unclassified. We define an indication for headers that cannot be classified correctly by a given encoding. We denote this unique action by '?' and by $\mathcal{A}^*$ the generalized set of actions $\mathcal{A}^* = \mathcal{A} \cup \{\text{'?'}\}$. We would look for a classifier that for every header either returns the correct classification value or the indication '?'. Upon receiving such an indication, the classification can be completed, e.g. by using a slower module, in mechanism similar to memory caching.

We can now define the main optimization problem, where we look for an encoding that obtains a maximal approximation ratio while returning the action of '?' for all headers that are not classified correctly.

PROBLEM 2 (CACHED CLASSIFICATION). *For a given classification function α, a header distribution P and a given number of prefix rules n, find a classifier C^ϕ with at most n rules that obtains a maximal approximation ratio while satisfying $\forall x \in \{0,1\}^W$,*

$$(\phi(x) = \alpha(x)) \ OR \ (\phi(x) = \text{'?'}).$$

We denote by $\mathcal{H}(n, \alpha, P)$ the optimal (maximal) approximation ratio that can be obtained while satisfying this additional condition and we say that an encoding that achieves this is a cache-optimal encoding. In the context of this problem, we define the quantity $1 - R_P(\alpha, \phi)$ as the **cache miss ratio**. This is the fraction of headers mapped to '?'.

Note that for both problems the rules in an optimal encoding are not necessarily a subset of the rules in an exact classifier.

3.2 A Real-Life Illustrative Example

A classifier with prefix rules is often represented as a labeled binary prefix tree. Each node of the tree corresponds to a prefix rule, given by the transition bits along the path from the root node. A node that corresponds to a rule in the classification is labeled with the rule action. Fig. 2 shows an example of a binary tree which is a small branch of the tree for a real classifier. The left arc corresponds to a 0 bit transition, and the right to a bit of 1. Fig. 2(a) shows the ORTC compressed prefix tree. Here, ORTC requires 10 rules in total and guarantees 100% classification. The actions appear with labels in the tree and the probability of a header to have a longest match in a rule according to a real-life trace is illustrated. These probabilities rely on the header distribution of the captured traffic traces.

In Fig. 2(b) the probability that a given header is included within a given subtree appears next to the subtree such that in each subtree all headers have a match in the same rule. Subtrees without such numbers have a negligible probability. The result shown for optimal Approximate Classification with 3 rules can classify correctly 97.84% of the packets, and obtains false classifications for the rest. The subtrees from which headers are not classified correctly are drawn with dotted circles. The corresponding encoding has the prefix rules $01101/5 \rightarrow 0$, $1100/4 \rightarrow 1$ and $-/0 \rightarrow 2$. Fig. 2(c) shows the results of optimal Cached Classification with 7 rules. It can correctly classify 98.52% of the packets. It returns '?' for headers that cannot be classified correctly and there is no false classification. Table 1 shows for both schemes how the approximation ratio increases by allowing more rules.

4. PROPERTIES

Before solving these two problems, we would like to discuss some of their properties. First, to better understand the expected performance of solutions for the problems we discuss the values of the optimal approximation ratio in each of the problems.

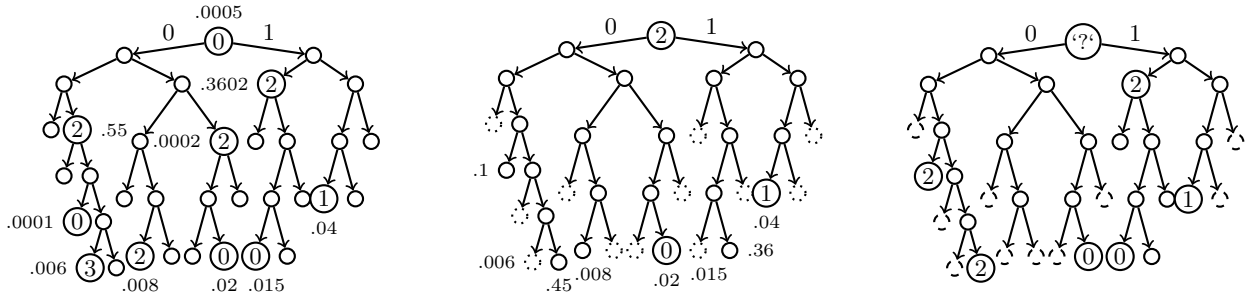
The first observation compares the optimal values of the approximation ratios of the two problems for the same number of rules. Intuitively, the requirement in the Cached Classification problem for a special indication on the headers that cannot be classified correctly results in a lower optimal ratio than that of the Approximate Classification problem.

OBSERVATION 1. *For (n, α, P), the optimal approximation ratio of the Approximate Classification problem equals at least the optimal approximation ratio of the Cached Classification problem and the ratios are equal only if an exact representation exists. I.e.,*

$$\mathcal{H}(n, \alpha, P) < \mathcal{G}(n, \alpha, P) < 1 \ or$$
$$\mathcal{H}(n, \alpha, P) = \mathcal{G}(n, \alpha, P) = 1.$$

Indeed, in some cases $\mathcal{G}(n, \alpha, P)$ can be much larger than $\mathcal{H}(n, \alpha, P)$. Consider for instance a classifier (with a classification function α) that maps a single header with an arbitrarily small but positive probability to the action 1 and all other headers to the action 2. Assume a default action that is not one of these two actions. For $n = 1$, an Approximate Classification encoding of $-/0 \rightarrow 2$ classifies almost all headers correctly and $\mathcal{G}(n = 1, \alpha, P)$ is close to 1. Any Cached Classification encoding with a single rule must be $-/0 \rightarrow$ '?'. This encoding has an approximation ratio of $\mathcal{H}(n = 1, \alpha, P) = 0$.

Before presenting more observations, we define the popularity of a prefix rule in a classifier.



(a) The minimal exact classifier requires 10 rules computed by ORTC [20]. The rule popularities are also drawn.

(b) Optimal *Approximate Classification* given 3 rules, $\mathcal{G}(3, \alpha, P) = 97.84\%$. The leaf probabilities are drawn (with negligible probabilities of the leaves without values). Headers in dotted leaves are incorrectly classified.

(c) Optimal *Cached Classification* given 7 rules, $\mathcal{H}(7, \alpha, P) = 98.52\%$. Headers matching the dashed leaves cannot be classified.

Figure 2: An illustrative example of a branch of a real classifier. The action of a prefix rule is drawn as a node label.

DEFINITION 7. *The* **prefix rule popularity** *of a rule in a classifier C is defined as the sum of the probabilities of all headers that have a longest match with the rule. For a rule $S^j \rightarrow a^j$ we denote this popularity by p^j. Formally,*

$$p^j = \sum_{x \in \{0,1\}^W | S^j \ LPM \ x} p_x \ ,$$

where p_x denotes the probability for a header x to appear.

See also Fig. 2(a) as an example of prefix rule popularities drawn next to the nodes along with the probabilities of each leaf drawn on Fig. 2(b).

We now present a general property of the classifiers. Intuitively, the contribution of a rule to the approximation ratio is limited by its popularity.

LEMMA 1. *Let C^ϕ be a classifier. By removing a rule $S^j \rightarrow a^j$ from the classifier the approximation ratio of the Approximate Classification problem is decreased by at most p^j, where p^j denotes the rule popularity according to Definition 7.*

The next observation suggests a lower bound for the optimal approximation ratio for the first problem. The bound is obtained as the ratio achieved by a classifier with a subset of the rules in the input classifier. Without adding additional rules, a header with a longest match in the input classifier in a selected rule will have such a match also in the smaller classifier.

THEOREM 1. *Let $C^\psi = (S^1 \rightarrow a^1, ..., S^{n_0} \rightarrow a^{n_0})$ be a classifier with a minimal number n_0 of prefix rules for a given classification function α. Assume a header distribution P. Let p^i be the prefix rule popularity of $S^i \rightarrow a^i$ when the rules are ordered in a non-increasing order of their popularities. By relying on a classifier composed of the n rules with the largest popularities, the optimal approximation ratio satisfies*

$$\mathcal{G}(n, \alpha, P) \geq \sum_{j \in [1,n]} p^j + 1 - \sum_{j \in [1,n_0]} p^j \geq n/n_0.$$

This intuitively leads to a simple greedy algorithm for Approximate Classification which selects the n rules with highest rule popularity (see also Table 1(a) as an example).

We can also derive a lower bound for the optimal approximation ratio for the Cached Classification problem based on the ratio obtained by such a classifier.

OBSERVATION 2. *Let n_0 denote the minimum number of rules needed for the exact classifier of classification function α. Let us sort the prefix rules according to a non-increasing order of their lengths. For $n \in [1, n_0]$, the optimal approximation ratio of the Cached Classification problem satisfies*

$$\mathcal{H}(n, \alpha, P) \geq \Sigma_{j \in [1, n-1]} p^j.$$

Similarly this leads to a simple greedy algorithm for Cached Classification which selects the $n-1$ longest rules and adds a last rule of $-/0 \rightarrow$ '?' (see also Table 1(b) as an example).

5. APPROXIMATE CLASSIFICATION

In this section we present algorithms that obtain optimal solutions for the Approximate Classification problem as defined in Problem 1. We see the Approximate Classification scheme as a baseline scheme that can help with the understanding of the Cached Classification which is the main scheme of the paper. We first assume a simple case in which the given classifier is represented by rules with distinct actions and present an immediate solution for this case. Later, we describe a more general algorithm of a classifier with arbitrary actions. This algorithm relies on dynamic programming.

5.1 The Case of Distinct Actions

We now consider the case in which all rules of a classifier have distinct actions and that these actions differ from the default action a^-. Our first observation is that removing the j^{th} rule decreases the approximation ratio by exactly p^j; all traffic that previously matched this rule is now not classified correctly. Thus to maximize the approximation ratio, we should include the rules with the highest popularity.

OBSERVATION 3. *Let C be a classifier with n_0 rules with distinct actions that also differ from the default action a^-.*

rule	rule pop.
$011/3 \to 2$	.55
$10/2 \to 2$	.3602
$1100/4 \to 1$	.04
$01101/5 \to 0$	.02
$10100/5 \to 0$	.015
$01010/5 \to 2$	.008
$001110/6 \to 3$	.006
$-/0 \to 0$	.0005
$001/3 \to 2$	.0002
$00110/5 \to 0$	.0001

n	greedy approx. ratio	optimal $\mathcal{G}(n, \alpha, P)$
1	0.55	0.55
2	0.9102	0.9584
3	0.9502	0.9784
4	0.9702	0.9934
5	0.9852	0.9994
6	0.9932	0.9996
7	0.9992	0.9997
8	0.9997	0.9998
9	0.9999	0.9999
10	1	1

(a) Approximate Classification

rule	rule pop.
$001110/6 \to 3$	.006
$01101/5 \to 0$	.02
$10100/5 \to 0$	.015
$01010/5 \to 2$	.008
$00110/5 \to 0$	.0001
$1100/4 \to 1$	.04
$011/3 \to 2$	.55
$001/3 \to 2$	.0002
$10/2 \to 2$	.3602
$-/0 \to 0$	.0005

n	greedy approx. ratio	optimal $\mathcal{H}(n, \alpha, P)$
1	0	0
2	.006	0.45
3	.026	0.81
4	.041	0.91
5	.049	0.95
6	.0491	0.97
7	.0891	0.9852
8	.6391	0.9932
9	.6393	0.9993
10	.9995	1

(b) Cached Classification. In the greedy algorithm the last n^{th} rule is always $-/0 \to$ '?'.

Table 1: Illustration of the greedy algorithms for the two schemes on the example of Fig. 2: rules are considered in different orders based on their popularities and lengths. For a number of rules n the obtained greedy approximation ratio is compared with the optimal ratio.

For $n \in [1, n_0]$, an optimal approximation ratio for the Approximate Classification problem is composed of n rules with the highest popularity among the n_0.

5.2 Arbitrary Actions

We now describe a dynamic programming based algorithm to find an approximation optimal encoding, according to the header distribution P, for a classifier with arbitrary actions. Our solution calculates encodings for nodes in the tree such that each encoding comprises a limited number of rules and includes a specific last rule.

Let r be the root of the complete binary tree of 2^W leaves that includes all headers. For a node x (represented by a corresponding prefix) in the complete binary tree, we consider an encoding with a maximal number of rules $n \in \mathbb{N}^+$ satisfying that its last rule (among the n) is of the form $x \to a$ for an action $a \in \mathcal{A}$. We define the function $g(x, n, a)$ as the maximal ratio of headers from the subtree x that can be classified correctly by such an encoding. Let $\phi(x, n, a)$ be an encoding with the above properties that achieves this ratio.

The next lemma relates the optimal approximation ratio $\mathcal{G}(n, \alpha, P)$ to a value of the function $g(x, n, a)$. It relies on the value of the function for the root r with a specific number of rules and last action.

LEMMA 2. *The optimal approximation ratio satisfies $\mathcal{G}(n, \alpha, P) = g(r, n+1, a^-)$ where r is the root node and a^- is the default action. Likewise, an approximation-optimal encoding is given by the first n rules in $\phi(r, n+1, a^-)$.*

We start by setting the values of $g(x, n, a)$ for a leaf (header) x assuming a classifier with a classification function α. Since there exists an encoding with n rules all of the form $x \to \alpha(x)$ we have

$$g(x, n, a) = p_x \quad \text{for } n \geq 1 \text{ if } a = \alpha(x).$$

Recall that p_x is the probability of a header to have the value of x. In addition,

$$g(x, 1, a) = 0 \text{ for } a \neq \alpha(x) \text{ and,}$$

$$g(x, n, a) = p_x \text{ for } n \geq 2.$$

We can have an encoding with two rules $(x \to \alpha(x), x \to a)$ that classifies x correctly for any action a.

More generally, for a given classifier consider the first matching rule for the 2^W headers represented by leaves in the binary tree of size 2^W. Consider a recursive partition of the set of leaves into halves until each subset of consecutive leaves contains headers that have a first match in the same rule. This partition divides the headers into disjoint *monochromatic* subtrees. As explained in [26] the number of these monochromatic subtrees equals at most $W \cdot n_0$ where n_0 is the number of rules in the classifier.

Let y be a prefix that represents such a monochromatic subtree, i.e. a subtree in which all headers have the first matching in the same rule. Let a^y denote the action of that rule. The above formulas for a leaf can be generalized for such a subtree as follows.

$$g(y, n, a) = p_y \quad \text{for } n \geq 1 \text{ if } a = a^y$$

where p_y is the probability of a header to be included in the subtree represented by y. Likewise,

$$g(y, 1, a) = 0 \text{ for } a \neq a^y \text{ and,}$$

$$g(y, n, a) = p_y \text{ for } n \geq 2.$$

The next lemma suggests a recursive formula for the value of $g(x, n, a)$ for a node x that is not a leaf. Similarly, the encoding $\phi(x, n, a)$ is calculated based on the two encodings for the left and right subtrees of x according to the value that is selected among the detailed cases.

LEMMA 3. *For a non-leaf node x and number of rules $n \geq 1$, the function $g(x, n, a)$ satisfies*
$g(x, n, a) = \max$

$$\left\{ \begin{array}{c} \max\limits_{m \in [1,n]} g(x_{\mathcal{L}}, m, a) + g(x_{\mathcal{R}}, n-m+1, a), \\ \max\limits_{m \in [1,n-1], a_1 \in \mathcal{A}} g(x_{\mathcal{L}}, m, a_1) + g(x_{\mathcal{R}}, n-m, a_1) \end{array} \right\},$$

where $x_{\mathcal{L}}$ and $x_{\mathcal{R}}$ are the left child and the right child of the node x.

The algorithm works as follows. Based on the rules in the given classifier, we divide the complete binary tree into monochromatic subtrees. After setting the function values for the corresponding nodes, we calculate based on the recursive formulas from Lemma 3 the function values and the corresponding encodings for internal nodes. An optimal encoding is obtained by Lemma 2 as the encoding for specific parameter values for the root of the complete binary tree.

THEOREM 2. *The described algorithm obtains an optimal solution for the Approximate Classification problem.*

Theorem 3 describes the time and space complexity of the algorithm. This analysis simply relies on the above description of the algorithm.

THEOREM 3. *The Approximate Classification problem can be optimally solved in $O(W \cdot n_0 \cdot |\mathcal{A}| \cdot n^2)$ time and $O(W^2 \cdot n_0 \cdot |\mathcal{A}| \cdot n^2)$ space, where n_0 is the number of rules in an exact encoding of the classifier.*

The linear dependency of the time complexity and the quadratic dependency of the memory complexity in the header width W guarantee that the algorithm remains practical also for IPv6.

6. CACHED CLASSIFICATION

In this section we present an algorithm that obtains an optimal solution for the Cached Classification problem as defined in Problem 2. This algorithm is also based on dynamic programming.

Recall that we define the generalized set of actions $\mathcal{A}^*$ as $\mathcal{A}^* = \mathcal{A} \cup \{\text{`?'}\}$. Here, we only consider encodings that for any header $x$ either return the correct action $\alpha(x)$ or the action `?'. We define $h(x, n, a)$ as the maximal ratio of correctly classified headers in such an encoding with $n \in \mathbb{N}^+$ rules with a last rule of $x \to a$. In order to avoid illegal encodings, we use the function value of $-\infty$ if there does not exist an encoding that satisfies the above requirements. We also denote by $\psi(x, n, a)$ an example of an encoding that obtains this ratio.

As in the first problem we can deduce the optimal approximation ratio and an optimal encoding for the Cached Classification problem as follows.

LEMMA 4. *The optimal approximation ratio satisfies $\mathcal{H}(n, \alpha, P) = h(r, n+1, a^-)$ where r is the root node and a^- is the default action. Likewise, an approximation-optimal encoding is given by the first n rules in $\psi(r, n+1, a^-)$.*

Again, let y be a monochromatic subtree that its headers, with a total probability of p_y, all have a first match in the same rule with an action a^y. For this problem, the encodings $(y \to a^y)$ and $(y \to \text{`?'})$ are both legal but only the first of them classifies headers in y correctly. Accordingly,

$$h(y, 1, a) = p_y \text{ if } a = a^y \text{ and,}$$

$$h(y, 1, \text{`?'}) = 0.$$

On the contrary, an encoding of the form $(y \to a)$ is illegal if $a \neq a^y$ and $a \neq$ `?'. Thus

$$h(y, 1, a) = -\infty \text{ if } a \notin \{a^y, \text{`?'}\}.$$

For any action $a \in \mathcal{A}^*$ the encoding $(y \to a^y, y \to a)$ is legal and classifies all headers in y correctly. Thus

$$h(y, n, a) = p_y \text{ for } n \geq 2, a \in \mathcal{A}^*.$$

For a non-leaf node x the values of $h(x, n, a)$ and the corresponding encoding $\psi(x, n, a)$ should be calculated recursively as for g. Notice that if the two encodings for a left child $x_{\mathcal{L}}$ and a right child $x_{\mathcal{R}}$ are both legal, then the merged encoding for x is legal as well. Accordingly, the proof of the next lemma is similar to the proof of Lemma 3.

LEMMA 5. *For a non-leaf node x, and number of rules $n \geq 1$, the function $h(x, n, a)$ satisfies*
$h(x, n, a) = \max$

$$\left\{ \begin{array}{l} \max\limits_{m \in [1,n]} h(x_{\mathcal{L}}, m, a) + h(x_{\mathcal{R}}, n-m+1, a), \\ \max\limits_{m \in [1,n-1], a_1 \in \mathcal{A}^*} h(x_{\mathcal{L}}, m, a_1) + h(x_{\mathcal{R}}, n-m, a_1) \end{array} \right\}.$$

With the described changes in the initial values of the function, the dynamic programming algorithm is the same as for Approximate Classification, and its optimality can be also deduced from the above discussion.

THEOREM 4. *The described algorithm achieves an optimal solution for the Cached Classification problem.*

The time and space complexity of the algorithm is essentially the same as described in Theorem 3. Putting together Lemma 4 and 5 we have the following theorem.

THEOREM 5. *The Cached Classification problem can be optimally solved in $O(W \cdot n_0 \cdot |\mathcal{A}| \cdot n^2)$ time and $O(W^2 \cdot n_0 \cdot |\mathcal{A}| \cdot n^2)$ space, where n_0 is the number of rules in an exact encoding of the classifier.*

As stated in Observation 1, requiring an encoding for the Cached Classification problem to assign a special indication to every header that cannot be classified correctly has a cost of potential lower performance. In Section 9 we compare the optimal approximation ratios of the two problems.

7. MORE GENERAL CLASSIFIERS

In this section, we generalize our novel approach of lossy compression to additional types of classifiers.

7.1 Numerical Classification

We describe new metrics to capture the notion of similarity between classifiers. Then, we define new optimization problems for finding limited-size classifiers and explain how the previously mentioned algorithms can be modified to obtain optimal results for the new problems.

In the new described problems, we distinguish between two classifiers, even if they incorrectly classify the same set of headers, based on the exact values of the actions for the incorrectly classified headers. Assume a classifier in which the possible classification results (thus far called actions) are among a set of numerical values, i.e. $\mathcal{A} \subseteq \mathbb{R}$. Then, the difference $a_1 - a_2$ and the absolute difference $|a_1 - a_2|$ of two actions $a_1, a_2 \in \mathcal{A}$ are well defined. For instance, in such a numerical classification a header of a flow can be mapped to its required QoS level or to its allowed traffic rate.

The following problem generalizes the Approximate Classification problem. Here, we limit the encoder to classify a header to a value not smaller than the correct one.

PROBLEM 3 (ONE-SIDED APPROXIMATE). *For a classification function α, a header distribution P and a number of prefix rules n, find a classifier C^ϕ with at most n rules that obtains a maximal approximation ratio $R_P(\alpha, \phi)$ while satisfying $\forall x \in \{0,1\}^W$*

$$\phi(x) \geq \alpha(x).$$

To solve Problem 3, we define $b(y, n, a)$ as the maximal approximation ratio in an encoding with n rules for headers

in a subtree rooted by a node y such that the last rule is $y \to a$. Again, let p_y be the probability of a header to be included in the subtree represented by y and let $\mathcal{B}(n, \alpha, P)$ be the optimal value of the approximation ratio in this constrained problem. Our algorithm is based on the following lemma. The initial values of the function $b(y, n, a)$ for monochromatic trees enforce the restriction on the classification values for any header. The optimal approximation ratio again can be calculated based on the root node r.

LEMMA 6. *The function $b(y, n, a)$ satisfies*
(i) *For a monochromatic node y with a corresponding action a^y: $b(y, 1, a^y) = p_y$, $b(y, 1, a) = 0$ for $a > a^y$, $b(y, 1, a) = -\infty$ for $a < a^y$. Likewise, $b(y, n, a) = p_y$ for $n \geq 2, a \in \mathcal{A}$.*
(ii) $\mathcal{B}(n, \alpha, P) = b(r, n+1, a^-)$ *and for a non-leaf node y, $b(y, n, a) =$* max

$$\left\{ \begin{array}{l} \max\limits_{m\in[1,n]} b(y_{\mathcal{L}}, m, a) + b(y_{\mathcal{R}}, n-m+1, a), \\ \max\limits_{m\in[1,n-1],a_1\in\mathcal{A}} b(y_{\mathcal{L}}, m, a_1) + b(y_{\mathcal{R}}, n-m, a_1) \end{array} \right\}.$$

In Problems 4 and 5, our goal is to find a classifier that minimizes the average difference between the requested actions and the obtained one. Given a classifier with a classification function α, we define the dissimilarity of a classifier with a classification function ϕ as $\Delta_P(\alpha, \phi) = \sum_{x\in\{0,1\}^W} p_x \cdot |\alpha(x) - \phi(x)|$. While in the next problem, there are no constraints on the obtained actions for a specific header, in the later problem every header must be classified to a value not smaller than its corrected value.

PROBLEM 4 (UNCONSTRAINED DISSIMILARITY). *For a classification function α, a header distribution P and a given number of prefix rules n, find a classifier C^ϕ with at most n rules that minimizes the dissimilarity $\Delta_P(\alpha, \phi)$.*

PROBLEM 5 (ONE-SIDED DISSIMILARITY). *For a classification function α, a header distribution P and a given number of prefix rules n, find a classifier C^ϕ with at most n rules that minimizes the dissimilarity $\Delta_P(\alpha, \phi)$ while satisfying $\forall x \in \{0,1\}^W$ $\phi(x) \geq \alpha(x)$.*

Notice that Problem 4 and Problem 5 are minimization problems, unlike the previously described problems.

We derive dynamic programming based solutions also for these problems. We define $\mathcal{U}(n, \alpha, P), \mathcal{O}(n, \alpha, P)$ as the optimal (minimal) dissimilarity values that can be obtained for the last two problems with n rules given α and P.

To solve Problem 4, we define $u(y, n, a)$ as the minimal possible dissimilarity value obtained in an encoding for headers in a subtree rooted by a node y with n rules such that the last rule is $y \to a$. Again, let p_y be the probability of a header to be included in the subtree represented by y.

LEMMA 7. *The function $u(y, n, a)$ satisfies*
(i) *For a monochromatic node y with a corresponding action a^y: $u(y, 1, a) = |a - a^y| \cdot p_y$ and $u(y, n, a) = 0$ for $n \geq 2$.*
(ii) $\mathcal{U}(n, \alpha, P) = u(r, n+1, a^-)$ *and for a non-leaf node y, $u(y, n, a) =$* min

$$\left\{ \begin{array}{l} \min\limits_{m\in[1,n]} u(y_{\mathcal{L}}, m, a) + u(y_{\mathcal{R}}, n-m+1, a), \\ \min\limits_{m\in[1,n-1],a_1\in\mathcal{A}} u(y_{\mathcal{L}}, m, a_1) + u(y_{\mathcal{R}}, n-m, a_1) \end{array} \right\}.$$

Similarly we define the function $o(y, n, a)$ for Problem 5 and have the following.

LEMMA 8. *The function $o(y, n, a)$ satisfies*
(i) *For a monochromatic node y with a corresponding action a^y: If $a \geq a^y$ then $o(y, 1, a) = |a - a^y| \cdot p_y$ and if $a < a^y$ then $o(y, 1, a) = \infty$. In addition, $o(y, n, a) = 0$ for $n \geq 2$.*
(ii) $\mathcal{O}(n, \alpha, P) = o(r, n+1, a^-)$ *and for a non-leaf node y $o(y, n, a) =$* min

$$\left\{ \begin{array}{l} \min\limits_{m\in[1,n]} o(y_{\mathcal{L}}, m, a) + o(y_{\mathcal{R}}, n-m+1, a), \\ \min\limits_{m\in[1,n-1],a_1\in\mathcal{A}} o(y_{\mathcal{L}}, m, a_1) + o(y_{\mathcal{R}}, n-m, a_1) \end{array} \right\}.$$

The dynamic programming algorithms can be easily derived from the above formulas. The analysis of their time and memory complexities is the same as for the previously mentioned algorithms.

7.2 Two-Dimensional Classifiers

We briefly discuss how all the presented algorithms can be generalized for two-dimensional prefix classifiers, a popular class of classifiers in which rules are defined on two fields such as the source IP and the destination IP addresses. A two-dimensional prefix rule is composed of two one-dimensional prefixes and allows headers that match in both fields. In order for the LPM-based matching to be unambiguous, we assume as in [26] that the given classifier is consistent. In such a classifier, any two rules are either disjoint or nested, i.e. either the set of matching headers in one rule is a subset of the set in the second or these sets are disjoint.

With a limited number of allowed rules, our goal is to find a classifier that achieves a maximal approximation ratio based on a known distribution of the two-dimensional headers. For a prefix x in the first field and a prefix y in the second, we calculate an optimal encoding of the headers in the rectangle (x, y). Such a rectangle represents the Cartesian product of the two subtrees that correspond to the prefixes x, y in the two fields. The algorithms for all discussed problems can be generalized in a similar way. The key observation is that an optimal encoding for the rectangle is obtained by splitting it into two halves along one of the fields. A similar claim appears in [26] regarding exact representations of classifiers.

8. DISCUSSION

Dealing with incorrect classifications and cache misses: The choice of Approximate Classification vs. Cached Classification and the method for handling incorrect classification and cache misses depend on the application. For instance, with classification errors loops can occur in a routing application, and an application designed to filter illegal traffic might erroneously allow unwanted packets. Hence, Cached Classification would be more suitable for these applications. In some applications, e.g. load balancing among servers in a data center network, incorrect classification for a small fraction of the traffic might be tolerable. In the Cached Classification scheme, upon receiving '?' we have several choices how to obtain a correct classification. First, we can calculate the classification in a slower path, i.e. by accessing a second-level larger memory or, in software defined networks, by sending one packet header of a flow that cannot be classified to the network controller.

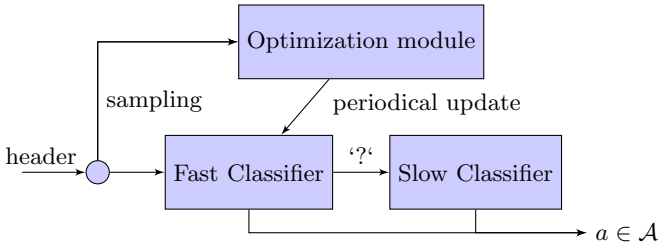


Figure 3: Block diagram of the prototype for the Cached Classification scheme.

Table 2: Description of FIBs examined with the number of distinct actions, the number of leaves and nodes in their original representations, and the number of ORTC rules.

FIB Name	#actions	#leaves	#nodes	#rules (ORTC)
SFR-HMS	27	235624	471247	71802
AS1221	4	261889	523777	94231
AS4637	3	105234	210467	35872
AS6447	36	375261	750521	160835
AS6730	186	336828	673655	140481
HBONE	195	284716	569431	107739
TAZ	4	150095	300189	49285

Generality of the solutions: We have described algorithms that obtain the optimal classifier with a limited number of rules for the Approximate Classification problem, the Cached Classification problem, and for additional problems related to numerical classifiers. The algorithms differ in the initial values of the recursive formulas for the monochromatic subtrees. In all problems, the function value of a node for a possible limited-size encoding is given by the sum of the values of its two subtrees. As mentioned, the number of rules in an encoding achieved by combining two encodings for two adjacent subtrees does not depend on a specific metric. Let the function $I(\cdot)$ be the indicator function that takes the value of 1 if the condition that it receives as an argument is satisfied, and 0 otherwise. The algorithms can be generalized to any metric in which the value of an encoding that implements a function ϕ is given by $\sum_{x\in\{0,1\}^W} \mathcal{F}(\alpha(x), \phi(x), p_x)$ for an arbitrary function $\mathcal{F}$, and the constraints of the function ϕ can be also expressed based on the values for each header. In such cases, the similarity of the functions is separable for the different headers. In particular, for the described problems we have for Approximate Classification $\mathcal{F}(\alpha(x), \phi(x), p_x) = p_x \cdot I(\alpha(x) = \phi(x))$, for Cached Classification it can be $p_x \cdot I(\alpha(x) = \phi(x)) - 1 \cdot I(\phi(x) \notin \{\alpha(x), \text{'?'}\})$. Here, an incorrect classification of a single header decreases the function value for the complete binary tree by at least one and guarantees that its value will not be positive. Likewise, for One-Sided Approximate the function can be $p_x \cdot I(\alpha(x) = \phi(x)) - 1 \cdot I(\alpha(x) > \phi(x))$, for Unconstrained Dissimilarity the function is $p_x \cdot |\alpha(x) - \phi(x)|$ and for One-Sided Dissimilarity it is $p_x \cdot |\alpha(x) - \phi(x)| + I(\alpha(x) > \phi(x))$. This family of algorithms can be presented as an extension of the algorithm from [26] for the special case of exact encoding. An exact encoding can be obtained with a function $\mathcal{F}$ of the form $1 - I(\alpha(x) = \phi(x))$. Then an exact encoding exists for a value of n for which a zero value of $\mathcal{F}$ exists for the complete binary tree.

We have assumed that the classification function α in the input is represented by an (exact) prefix classifier. The algorithms can find optimal solutions with prefix rules also for an arbitrary function α. However, in this case the number of monochromatic trees can be much larger resulting in larger time complexity of the algorithms.

9. EXPERIMENTAL RESULTS

We performed extensive simulations to validate the proposed approaches on a workstation with a 2.50GHz Intel Core i5 CPU. Since the effectiveness of the proposed approach depends on the correlation between the header distribution and the input classifiers, ideally they should be collected together at the same time in the same network component. We could arrange one such measurement in a campus network. In addition to the documentation of the corresponding Forwarding Information Base (FIB), a full day traffic capture was performed on December 22, 2014. The measured link was a 10Gigabit Ethernet port of a Cisco 6500 Layer-3 switch which transfers the traffic of two buildings on a campus site to the core layer of the local network. We refer to this measurement as the BME FIB and trace. As additional classifiers, we used 7 access and core FIB instances taken from [21], as summarized in Table 2. We used real Internet traffic traces to calculate the header distribution measured at four Tier 3 Internet Service Providers (ISP) on 8 different routers. Each trace was captured for 2-4 minutes with WireShark inlcuding at least 2 million packets. For each FIB we used each traffic trace to compute the exact header distribution, that in total resulted 7×8 problem instances. Note that, in practice the exact header distribution is not known in advance, but instead it can be estimated according to some historical data.

We implemented a prototype of the Cached Classification scheme. The prototype has a *fast classifier* with a small memory and a *slower classifier* without memory limits. The fast classifier corresponds to a wire speed hardware device, while the slow classifier to a software-based component. Note that, the hardware component can be 10 to 100 times faster than the second component. Fig. 3 shows the block diagram of the prototype. A packet first reaches the fast classifier and is either classified correctly, or the indication '?' is obtained and the packet is sent also to the slower classifier. The prefix rules in the fast classifier are updated periodically once in an *update period*. At the end of each period a new set of rules is computed for the fast classifier according to the measured header distribution in the current period. The header distribution is measured by sampling the incoming packets. We examined several *sampling rates* $1 : x$, for which one among x consecutive packets is used to compute the header distribution. The new rule set is loaded to the fast classifier for the next period after some delay. This delay is modeled as the computational time of the algorithm plus a fixed *loading time* that represents the time it takes to load a new set of rules in the fast classifier.

Fig. 4 shows the probability of a header to be included in each of the leaves in the ORTC representation of the BME FIB according to the BME traffic trace truncated to different time periods. See also Fig. 2 as an illustration, where the probability of each leaf is drawn next to the leaves. We sort the leaves in non-increasing order of the probabilities and

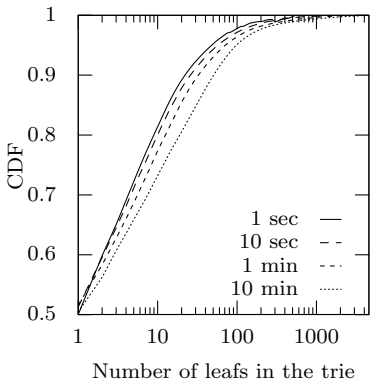
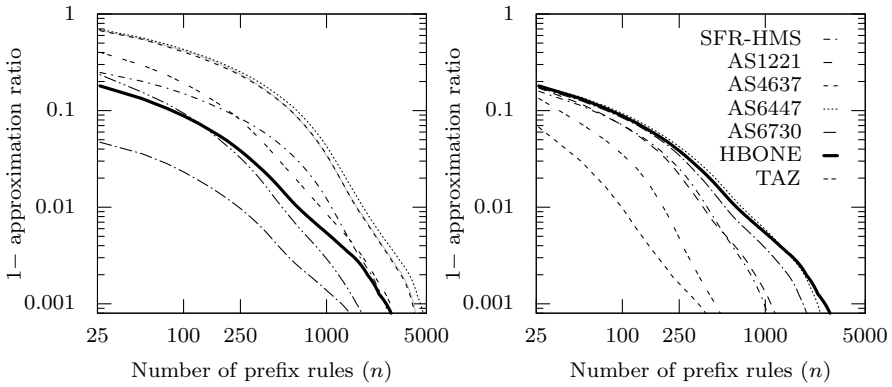


Figure 4: The probabilities to be included in a leaf in the BME FIB according to the BME traffic trace.



(a) Various traffic traces on the same HBONE FIB for Cached Classification.

(b) Various FIBs on the same traffic trace.

Figure 5: The sensitivity of the input data.

Fig. 4 shows the empirical cumulative distribution function (CDF) as a function of the length of the truncated traffic trace. Note that roughly 50% of the traffic is mapped to a single leaf. This illustrates how biased is the distribution we are facing in the classification process.

9.1 FIB Caching at Tier 3 Networks with Exact Header Distribution

On the axes of Fig. 5-7 the logarithmic scale is used. We plot 1 − approximation ratio, which is also called the *error ratio* or the *cache miss ratio* for the Approximate Classification or the Cached Classification problems, respectively.

Fig. 5(a) shows the results of Cached Classification of HBONE FIB where the prefix popularities are computed with 8 different traffic traces. As detailed in Table 2, 107K rules are required for exact classification with ORTC, and surprisingly with 25 rules an approximation ratio of 30-95% is reached for the 8 traces. A ratio of 99% requires 250-1800 rules, and that of 99.9% 1300-4500 rules. On average, with exact knowledge of the header distribution, roughly 1% (1077) of the rules required by ORTC was sufficient to classify 99% of the traffic by the fast classifier.

In Fig. 5(b) all the FIBs were evaluated with a single traffic trace which is one of the 8 traces mentioned above. This trace has $2,206,097$ packets captured in 163 sec in a rate of $13,300$ packet/sec on backbone link with an average bandwidth of 105 MBit/sec. The same FIB and traffic trace pair is drawn with thick lines on Fig. 5(a) and (b). Here, an approximation ratio of 99% requires 100-700 rules for the same traffic trace depending on the FIB.

Fig. 6 shows the average error ratio for Cached and Approximate Classification over all the FIBs and traffic traces. It is an average of the 7×8 instances. The 95% confidence interval is also plotted as a shadow of the curves of the optimal algorithms. On average Cached Classification required 2.77 times more rules than Approximate Classification for the same approximation ratio. This factor decreases as we have a larger approximation ratio and the same number of rules is required to for an approximation ratio of 1. The figure also shows the results of the greedy heuristics introduced in Section 4 (see also Table 1 as an example). The

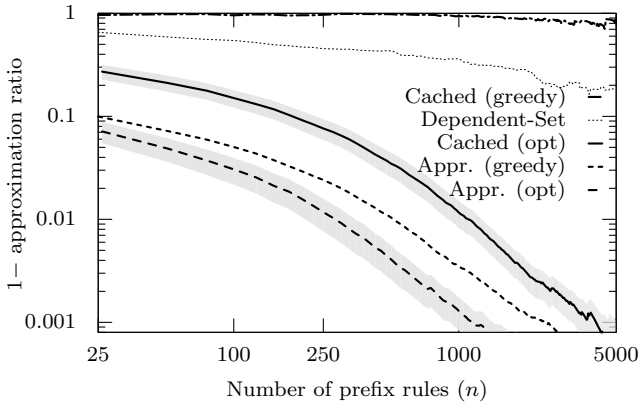
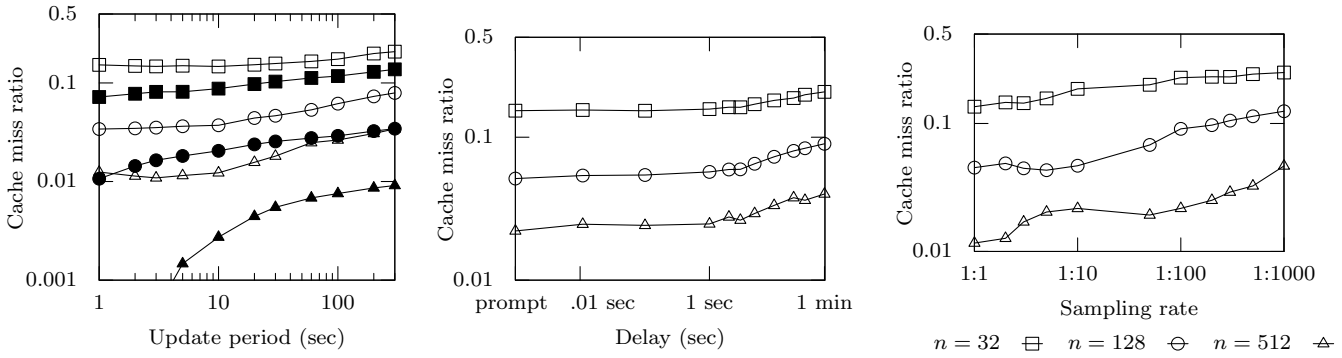


Figure 6: The error ratio and the cache miss ratio vs. the number of rules. The average over 7×8 problem instances with all FIBs and data traces. The 95% confidence interval is also plotted.

greedy algorithm for Approximate Classification provides a decent performance compared to the optimal encoding obtained by the dynamic program; however, the performance of the corresponding greedy algorithm for Cache Classification is not acceptable. The figure also shows the results of the algorithm Dependent-Set from [27] that also relies on rule caching. It is important to indicate that their solution is more general and supports also non-prefix classifiers. We used the ORTC rule set as the input to that algorithm.

9.2 The Effect of Imprecision in the Header Distribution

The described prototype has 3 parameters: the update period, the sampling rate and the loading time. Fig. 7(a) shows the approximation ratio as a function of the length of the update period for the BME trace and FIB. We used 32, 128 and 512 for the rule limit of the fast classifier. As a reference we also added the theoretical optimum for the approximation ratio, which is computed with the exact header



(a) For different size of classifiers and update periods (sampling 1:10, load time 0.1 sec). The filled marks shows the results using the exact header distribution.

(b) For different delays (update period 30sec, sampling 1:10). Prompt means the new rule set is computed and loaded in zero time.

(c) For different sampling rate (load time 0.1 sec, update period 5 sec).

Figure 7: The performance of the classifier prototype on BME FIB and trace.

distribution and shown with filled marks. In general longer update periods always results in lower approximate ratio.

We also investigated the effect of the load time of the fast classifier. We model the delay as the sum of the rule computation time and the time it takes to load a new set of rules to the fast classifier. Fig. 7(b) shows the approximation ratio as a function of the delay. Longer delays result in performance degradation as the header distribution changes in time. In our observation the prototype was little sensitive to the time of loading a new rule set to the fast classifier. Fig. 7(c) illustrates how the sampling rate affects the performance. Larger but reasonable cache miss ratios can be observed also for low sampling rates.

10. RELATED WORK

Compression of packet classifiers as well as of Forwarding Information Bases (FIBs) is a well studied problem. The problem was considered for a wide range of memories. The ORTC algorithm [20] obtains an optimal representation of a longest prefix match (LPM) classifier in the minimal possible number of prefix rules. A similar approach called FIB aggregation [28] suggests aggregating rules with the same action. Similar techniques are described in [29, 30]. Entropy bounds on the size of an LPM-based classifier and algorithms to obtain them were presented in [21]. [14] discussed how to reduce the width of a classifier by eliminating some of its fields. Codes for fixed-width memories have been described in [31, 32].

In particular, the problem of dealing with the limited size of TCAMs has been well studied. A wide range of memory-efficient representations of classifiers in TCAMs have been suggested [33–37]. For instance, the compression can be achieved by eliminating redundant rules [13], by learning the interactions between different rules [14, 15], or by performing block permutation [16]. There is a special interest in developing efficient TCAM coding schemes for range-based classification rules [17–19, 38–40].

The concept of using slow and fast classifiers for rule caching is not new [9, 10]. In an LPM-based classification, the existence of a matching rule for an incoming packet among the set of cached rules does not necessarily mean that this is the correct rule, since there might exist a longer matching rule among the non-cached rules. [41] described schemes for selecting a subset of rules that avoids this phenomenon, known as the cache hiding problem. [42] tackled the same problem by introducing rule caching for fast line cards wakeup. [27] described a system that caches the most popular rules in a small TCAM while handling the others in software. A recent approach suggested distributing the rules of a classifier among several limited-size TCAMs in the network [43–45].

11. CONCLUSIONS

In this paper, we have described a lossy compression approach for limited-size classification modules, in particular TCAMs. We have presented different similarity metrics for classifiers and developed algorithms that find optimal classifiers under different constraints. In particular, we have presented a scheme in which a special indication is always returned for headers that cannot be classified correctly. Then, a correct classification can be achieved by accessing the network controller or another memory level. We have explained how the approach can be applied to a wide range of classifiers within different modules. Extensive experiments showed a significant reduction in the size of real classifiers based on real campus traffic.

12. ACKNOWLEDGMENT

We thank our shepherd Srinivas Kadaba, the anonymous reviewers, Ronaldo A. Ferreira, Jennifer Gossels, Isaac Keslassy, Gabor Rétvári, Jennifer Rexford, Muhammad Shahbaz for their helpful suggestions as well as to Péter Megyesi for setting up the measurements.

APPENDIX

Proof of Observation 1: An exact representation obtains an approximation ratio of 1 for both problems. If such an exact representation does not exist ($n < n_0$), consider an optimal encoding for the second problem. It must include at least one rule with '?' that matches at least one header. Based on this encoding, we can simply obtain a solution for the first

problem that achieves a larger approximation ratio. To do so, we replace the action of '?' in such a rule by an action in $\mathcal{A}$ that corresponds to one of the headers with a first match in that rule. Such a change does not affect any header that was originally classified correctly, while at least one header previously mapped to '?' is now classified to its required action. □

Proof of Lemma 1: Any header that had a longest match in one of the rules other than $S^j \rightarrow a^j$ will have a longest match in the same rule after removing rule $S^j \rightarrow a^j$. □

Proof of Theorem 1: In such a classifier, a header that was classified correctly by having a longest match in a selected rule will have again a longest match in the same rule and will be classified correctly again. In addition, all headers that had no much in any of the n_0 will again have no much in the subset of rules. Such rules are mapped to the default action in both cases. The last lower bound is deduced by a simple lower bound on the average of the largest n popularities and the consideration of the probability that a header does not match any rule. □

Proof of Observation 2: Consider an encoding with n rules $S_1 \rightarrow a_1, \ldots, S_{n-1} \rightarrow a_{n-1}, -/0 \rightarrow$ '?' composed of the $n-1$ longest rules in the encoding of C^α and a last default rule that returns '?'. It classifies correctly all headers matching one of these $n-1$ longest rules in the exact encoding with n_0 rules and therefore achieves an approximation ratio of $\Sigma_{i\in[1,n-1]}p^i$. This encoding is legal since it returns '?' for any other header. □

Proof of Lemma 2: Consider an encoding $\phi(r, n+1, a^-)$ that obtains the approximation ratio $g(r, n+1, a^-)$. In such an encoding the last rule of the form $r \rightarrow a^-$ is redundant since a^- is the default action. By eliminating this rule we can have an encoding of n rules that achieves the same ratio and therefore $\mathcal{G}(n, \alpha, P) \geq g(r, n+1, a^-)$. Likewise, for any encoding with n rules that obtains $\mathcal{G}(n, \alpha, P)$ we can add a rule of the form $r \rightarrow a^-$ while still obtaining the same approximation ratio. This is a legal encoding for $g(r, n+1, a^-)$. Thus we also have that $\mathcal{G}(n, \alpha, P) \leq g(r, n+1, a^-)$ and the equality is satisfied. □

Proof of Lemma 6: For a monochromatic node y an encoding of the form $y \rightarrow a$ is legal if $a \geq a^y$ and achieves an approximation ratio greater than zero only if $a = a^y$. Likewise, a legal encoding for a non-leaf node y can be combined by the merging of legal encodings for the two subtrees regardless of the specific optimization function. □

Proof of Lemma 7: The proof is similar to the proof of the previous lemma with the following change: For a monochromatic node y the encoding $y \rightarrow a^y$ with a single rule has a dissimilarity of 0, while an encoding of the form $y \rightarrow a$ for $a \neq a^y$ has a dissimilarity of $|a - a^y| \cdot p_y$. □

Proof of Lemma 8: Again, the proof is similar to the previous proofs. To avoid an illegal encoding of the form $y \rightarrow a$ for the monochromatic node y when $a < a^y$, we set the value of the function $o(y, 1, a)$ to be ∞. □

A. REFERENCES

[1] Igor Gashinsky. Datacenter scalability panel. In *North American Network Operators Group, NANOG 52*, 2011.

[2] Dave Meyer, Lixia Zhang, and Kevin Fall. Report from the IAB Workshop on Routing and Addressing. In *RFC 4984, IETF*, 2007.

[3] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. In *ACM SIGCOMM*, 2013.

[4] Kostas Pagiamtzis and Ali Sheikholeslami. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-State Circuits*, 41(3):712–727, 2006.

[5] Yadi Ma and Suman Banerjee. A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification. In *ACM SIGCOMM*, 2012.

[6] Nadi Sarrar, Steve Uhlig, Anja Feldmann, Rob Sherwood, and Xin Huang. Leveraging Zipf's law for traffic offloading. *Computer Communication Review*, 42(1):16–22, 2012.

[7] Chuck Fraleigh, Sue Moon, Bryan Lyles, Chase Cotton, Mujahid Khan, Deb Moll, Rob Rockell, Ted Seely, and S. C. Diot. Packet-level traffic measurements from the Sprint IP backbone. *IEEE Network*, 17(6):6–16, 2003.

[8] Marina Fomenkov, Ken Keys, David Moore, and KC Claffy. Longitudinal study of Internet traffic in 1998-2003. In *WISICT*, 2004.

[9] Huan Liu. Routing prefix caching in network processor design. In *ICCCN*, 2001.

[10] Bryan Talbot, Timothy Sherwood, and Bill Lin. IP caching for terabit speed routers. In *IEEE Globecom*, 1999.

[11] David Huffman. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(9), 1952.

[12] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.

[13] Alex X. Liu, Chad R. Meiners, and Yun Zhou. All-match based complete redundancy removal for packet classifiers in TCAMs. In *IEEE Infocom*, 2008.

[14] Kirill Kogan, Sergey I. Nikolenko, Ori Rottenstreich, William Culhane, and Patrick Eugster. Exploiting order independence for scalable and expressive packet classification. *IEEE/ACM Trans. Netw.*, 2015.

[15] Eric Norige, Alex X. Liu, and Eric Torng. A ternary unification framework for optimizing TCAM-based packet classification systems. In *ACM/IEEE ANCS*, 2013.

[16] Rihua Wei, Yang Xu, and H. Jonathan Chao. Block permutations in boolean space to minimize TCAM for packet classification. In *IEEE Infocom Mini-Conference*, 2012.

[17] Hao Che, Zhijun Wang, Kai Zheng, and Bin Liu. DRES: Dynamic range encoding scheme for TCAM coprocessors. *IEEE Trans. Computers*, 57(7):902–915, 2008.

[18] Anat Bremler-Barr and Danny Hendler. Space-efficient TCAM-based classification using Gray coding. *IEEE Trans. Computers*, 61(1):18–30, 2012.

[19] Yeim-Kuan Chang, Chun-I. Lee, and Cheng-Chien Su. Multi-field range encoding for packet classification in TCAM. In *IEEE Infocom Mini-Conference*, 2011.

[20] Richard Draves, Christopher King, Srinivasan Venkatachary, and Brian Zill. Constructing optimal IP routing tables. In *IEEE Infocom*, 1999.

[21] Gábor Rétvári, János Tapolcai, Attila Korösi, András Majdán, and Zalán Heszberger. Compressing IP forwarding tables: towards entropy bounds and beyond. In *ACM SIGCOMM*, 2013.

[22] Gregory K. Wallace. The JPEG still picture compression standard. *Commun. ACM*, 34(4):30–44, 1991.

[23] Didier Le Gall. MPEG: A video compression standard for multimedia applications. *Commun. ACM*, 34(4):46–58, 1991.

[24] Michael Nilsson. *The audio/Mpeg Media Type*. RFC 3003, 2000.

[25] Recep Ozdag. Intel®Ethernet Switch FM6000 Series-Software Defined Networking. *Intel Coroporation*, 2012.

[26] Subhash Suri, Tuomas Sandholm, and Priyank Ramesh Warkhede. Compressing two-dimensional routing tables. *Algorithmica*, 35(4):287–300, 2003.

[27] Naga Katta, Omid Alipourfard, Jennifer Rexford, and David Walker. Rule-Caching algorithms for Software-Defined Networks. Technical report, Princeton University, 2014.

[28] Xin Zhao, Yaoqing Liu, Lan Wang, and Beichuan Zhang. On the aggregatability of router forwarding tables. In *IEEE Infocom*, 2010.

[29] Zartash Afzal Uzmi, Markus Nebel, Ahsan Tariq, Sana Jawad, Ruichuan Chen, Aman Shaikh, Jia Wang, and Paul Francis. SMALTA: practical and near-optimal FIB aggregation. In *ACM CoNEXT*, 2011.

[30] Yaoqing Liu, Xin Zhao, Kyuhan Nam, Lan Wang, and Beichuan Zhang. Incremental forwarding table aggregation. In *IEEE Globecom*, 2010.

[31] Ori Rottenstreich, Marat Radan, Yuval Cassuto, Isaac Keslassy, Carmi Arad, Tal Mizrahi, Yoram Revah, and Avinatan Hassidim. Compressing forwarding tables for datacenter scalability. *IEEE Journal on Selected Areas in Communications (JSAC)*, 32(1):138 – 151, 2014.

[32] Ori Rottenstreich, Amit Berman, Yuval Cassuto, and Isaac Keslassy. Compression for fixed-width memories. In *IEEE ISIT*, 2013.

[33] Chad R. Meiners, Alex X. Liu, and Eric Torng. TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs. In *IEEE ICNP*, 2007.

[34] Chad R. Meiners, Alex X. Liu, and Eric Torng. Bit Weaving: A non-prefix approach to compressing packet classifiers in TCAMs. *IEEE/ACM Trans. Networking*, 20(2):488–500, 2012.

[35] Anat Bremler-Barr, David Hay, Danny Hendler, and Boris Ferber. Layered interval codes for TCAM based classification. In *IEEE Infocom*, 2009.

[36] Chad R. Meiners, Alex X. Liu, and Eric Torng. Topological transformation approaches to TCAM-based packet classification. *IEEE/ACM Trans. Networking*, 19(1):237–250, 2011.

[37] Alexander Kesselman, Kirill Kogan, Sergey Nemzer, and Michael Segal. Space and speed tradeoffs in TCAM hierarchical packet classification. *J. Comput. Syst. Sci.*, 79(1):111–121, 2013.

[38] Ori Rottenstreich and Isaac Keslassy. On the code length of TCAM coding schemes. In *IEEE ISIT*, 2010.

[39] Ori Rottenstreich and Isaac Keslassy. Worst-case TCAM rule expansion. In *IEEE Infocom Mini-Conference*, 2010.

[40] Ori Rottenstreich, Isaac Keslassy, Avinatan Hassidim, Haim Kaplan, and Ely Porat. Optimal In/Out TCAM encodings of ranges. *IEEE/ACM Trans. Netw.*, 2015.

[41] Yaoqing Liu, Syed Obaid Amin, and Lan Wang. Efficient FIB caching using minimal non-overlapping prefixes. *Computer Communication Review*, 43(1):14–21, 2013.

[42] Tian Pan, Ting Zhang, Junxiao Shi, Yang Li, Linxiao Jin, Fuliang Li, Jiahai Yang, Beichuan Zhang, and Bin Liu. Towards zero-time wakeup of line cards in power-aware routers. In *IEEE Infocom*, 2014.

[43] Yossi Kanizo, David Hay, and Isaac Keslassy. Palette: Distributing tables in software-defined networks. In *IEEE Infocom*, 2013.

[44] Nanxi Kang, Zhenming Liu, Jennifer Rexford, and David Walker. Optimizing the "one big switch" abstraction in software-defined networks. In *ACM CoNEXT*, 2013.

[45] Minlan Yu, Jennifer Rexford, Michael J. Freedman, and Jia Wang. Scalable flow-based networking with DIFANE. In *ACM SIGCOMM*, 2010.